PATENT ABSTRACTS OF JAPAN

(11)Publication number : **07-182169**

(43)Date of publication of application : **21.07.1995**

(51)Int.Cl. **G06F 9/38**
G06F 9/30

(21)Application number : **05-327806**

(22)Date of filing : **24.12.1993**

(71)Applicant : **TOSHIBA CORP**

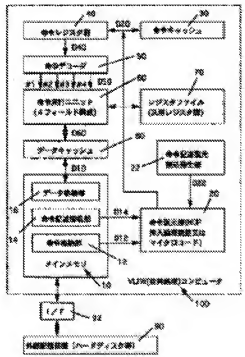
(72)Inventor : **ABE YAYOI**
TAKEUCHI YOICHIRO

(54) PARALLEL PROCESSING COMPUTER

(57)Abstract:

PURPOSE: To increase the free areas of a memory for a parallel processing type computer by reducing the size of the program to be loaded in the memory.

CONSTITUTION: A non-execution instruction NOP part is properly deleted out of a description part of a parallel processing instruction consisting of plural fields, so that the size of the instruction code part of a program to be loaded in a memory 10 is reduced. When the parallel processing instruction of the program is carried out, it is checked whether the compressed instruction code is equal to only a single instruction based on the contents of an instruction description information part 14. If the compressed instruction code is equal to only a single instruction, an instruction restoring part 20 fills the fields of parallel processing instructions excluding the relevant instruction with the NOP parts and restores the parallel processing instruction to be carried out. This restored instruction is once stored in an instruction cache 30. Then the restored parallel processing instructions stored in the cache 30 are carried out is sequence by an executing unit 60.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平7－182169

(43)公開日　平成７年(1995)７月21日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/38	３７０　Ｘ			
9/30	３５０　Ｆ			

審査請求　未請求　請求項の数５　ＯＬ　（全 11 頁）

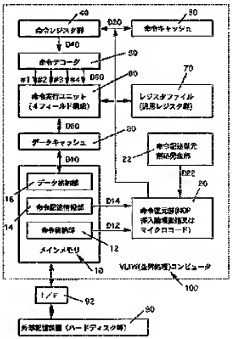
(21)出願番号　特願平5－327806

(22)出願日　平成５年(1993)12月24日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　安部　弥生
東京都府中市東芝町１番地　株式会社東芝府中工場内
(72)発明者　竹内　誠一郎
東京都府中市東芝町１番地　株式会社東芝府中工場内
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　並列処理型コンピュータ

(57)【要約】
【目的】並列処理型コンピュータにおいてメモリにロードされるプログラムサイズを小さくしてメモリのフリーエリアを増やす。
【構成】複数フィールドからなる並列処理命令の命令記述部から無実行命令ＮＯＰ部分を適宜取り除くことにより、メモリ１０にロードされるプログラムの命令コード部分のサイズを圧縮しておく。このプログラムの並列処理命令を実行する際に、圧縮された命令コードが１命令のみであるかどうかを命令記述情報部１４の内容から判断する。１命令のみであれば、命令復元部２０によりその命令以外の並列処理命令フィールドをＮＯＰで埋めて実行すべき並列処理命令を復元する。復元された命令は命令キャッシュ３０に一旦格納され、キャッシュ３０内の復元された並列処理命令が実行ユニット６０で順次実行される。



1

【特許請求の範囲】

【請求項1】　無実行命令以外の命令を含む情報を格納する命令格納部および無実行命令を付加するかどうかを示す情報を格納する命令記述情報部を記憶するメモリと；前記命令記述情報部の内容が無実行命令の付加を示す場合に、前記命令格納部の命令に無実行命令を付加して複数命令からなる命令コードを生成する命令生成手段と；複数の命令実行フィールドを持ち、これらの命令実行フィールドにおいて、前記命令生成手段で生成された命令コードを並列に実行する命令実行手段とを具備したことを特徴とする並列処理型コンピュータ。

【請求項2】　前記メモリは、無実行命令以外の命令を複数含む命令コードを格納する複数命令領域を持ち、前記命令生成手段は、前記命令記述情報部の内容が無実行命令の付加以外を示す場合に前記複数命令領域に格納された命令コードを取り出す命令取出手段を含み、前記命令実行手段が、前記命令生成手段により生成された命令コードとともに前記命令取出手段により取り出された命令コードを実行するように構成されることを特徴とする請求項1に記載の並列処理型コンピュータ。

【請求項3】　複数命令からなる並列処理命令コードが無実行命令以外の命令を1つだけ含む場合に、この並列処理命令コードから無実行命令を取り除くことによりこの並列処理命令コードのコードサイズを圧縮し、コードサイズが圧縮されたことを示す命令記述情報をこの圧縮された並列処理命令コードとともに保存し、前記命令記述情報が、保存された前記並列処理命令コードはコードサイズが圧縮されていることを示す場合に、この圧縮された命令コードに無実行命令を付加することにより並列処理命令コードを復元しながら実行するように構成したことを特徴とする並列処理型コンピュータシステム。

【請求項4】　命令記述情報部および命令格納部を具備し、並列処理命令コードから無実行命令部分を適宜取り除くことによりこの並列処理命令コードのコードサイズを圧縮し、このコードサイズが圧縮されたことに関する情報を前記命令記述情報部に記述し、このコードサイズが圧縮された命令コードを前記命令格納部に記述することを特徴とする並列処理型コンピュータシステム。

【請求項5】　命令記述情報部、命令格納部、命令記述復元割込部および命令復元部を具備し、並列処理命令コードから無実行命令部分を適宜取り除くことによりこの並列処理命令コードのコードサイズを圧縮し、このコードサイズが圧縮されたことに関する情報を前記命令記述情報部に記述し、このコードサイズが圧縮された命令コードを前記命令格納部に記述し、

2

前記命令記述情報部に記述された情報に基づき前記命令格納部に記述された圧縮命令コードに無実行命令を挿入することで元の並列処理命令コードを復元し、前記復元された元の並列処理命令コードを逐次実行することを特徴とする並列処理型コンピュータシステム。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、ベリーロングインストラクションワード（以下ＶＬＩＷと略記する）型などの並列処理型コンピュータにおいて、メインメモリにロードされるプログラムの命令コード部分のサイズを圧縮しておき、メインメモリにロードされたプログラムの各命令が実行される時点で圧縮された命令コードを復元するシステムに関する。

【０００２】【従来の技術】たとえばＶＬＩＷ型並列処理コンピュータにおいて実行されるシーケンシャルな命令コードには、命令記述以外の部分に無実行命令（以下ＮＯＰと略記する）が埋め込まれる。

【０００３】【発明が解決しようとする課題】並列処理プログラムにおけるシーケンシャルなコードでは、命令実行を行なわないＮＯＰが多くなるために、プログラムの機能の割に命令記述部のサイズが大きくなる。

【０００４】大規模なプログラムの場合、メインメモリにロードされたプログラム中のＮＯＰ部分がメモリを無駄食いするから、プログラム実行時に多くのメモリが必要となる。すると、プログラム実行中に使用できる残りのメモリ容量が少なくなり（つまりメモリが圧縮され）、必要な処理ができなくなるという問題が起こり得る。

【０００５】たとえばメインメモリ４メガバイトを装備したコンピュータにおいて、オペレーティングシステム（以下ＯＳと略記する）が起動したあとのメモリのフリーエリアが２．５メガバイトであったとする。このコンピュータのＯＳ上でＮＯＰを多く含むプログラムを起動したところメモリのフリーエリアが１．８メガバイトしか残らなかったとしたら　このプログラムで２メガバイトのフリーエリアを必要とする処理はけなうことができない（０．２メガバイトのメモリが不足）。

【０００６】とくにプログラム開発中において、デバッグ等のためにコードの大部分がシーケンシャル実行用になっていてプログラム中に存在するＮＯＰの割合が多い場合に、上記メモリ不足が起きやすい。このメモリ不足が起きた場合、コンピュータ本体のメモリを増設しない限り（そのコンピュータ自体およびＯＳがそのメモリ増設に対応していることが前提）、デバッグができず、プログラム開発が中断してしまう。

【０００７】またデバッグ終了後のプログラムに対しては　その起動後にユーザが使えるメモリのフリーエリア

サイズは大きいほど望ましい。この発明の目的は、メモリにロードされるプログラムサイズを小さくしてメモリのフリーエリアを増やすことのできる並列処理型コンピュータを提供することである。

【０００８】

【課題を解決するための手段】複数フィールドからなる並列処理命令の命令記述部から無実行命令ＮＯＰ部分を適宜取り除くことにより、メモリにロードされるプログラムの命令コード部分のサイズを圧縮しておく。このプログラムの並列処理命令を実行する際に、圧縮された命令コードが１命令のみであるかどうかを命令記述情報部の内容から判断する。１命令のみであれば、命令復元部により、その命令以外の並列処理命令フィールドをＮＯＰで埋めて実行すべき並列処理命令を復元する。復元された命令は命令キャッシュまたは命令バッファに一旦格納され、キャッシュまたはバッファ内の復元された並列処理命令が実行ユニットで順次実行される。

【０００９】

【作用】メモリにロードされるプログラムは圧縮されているから、圧縮された分だけメモリのフリーエリアは広くなっている。圧縮された命令はそのままでは実行できないので、実行前に除去されたＮＯＰを再挿入する復元が行なわれる。実行の度に（実行直前に）圧縮命令を逐次復元するようにしたから　並列処理命令の実行に差し障りはない一方で、メモリのフリーエリアをより広くできる。

【００１０】

【実施例】図１は、この発明の一実施例に係るＶＬＩＷ型並列処理コンピュータ１００の構成を示すブロック図である。このコンピュータ１００は、メインメモリ１０、命令キャッシュ３０、命令レジスタ群４０、命令デコーダ５０、命令実行ユニット６０、レジスタファイル（汎用レジスタ群）７０、およびデータキャッシュ８０を備えている。

【００１１】命令実行ユニット６０は、たとえば、最大４命令を並列処理するために４つの命令実行ユニット（フィールド＃１～＃４）を備えている。実行ユニット６０の各フィールド＃１～＃４では　レジスタファイル７０を利用しながら与えられた命令が別々に実行され、その実行結果がデータキャッシュ８０あるいはメモリ１００のデータ格納部１６に書き込まれるようになっている。

【００１２】コンピュータ１００で並列処理されるプログラムは、そのプログラムを使用するときに、外部記憶装置（ハードディスクなど）９０からインターフェイス（Ｉ／Ｆ）９２を介してメインメモリ１０に読み込まれる。

【００１３】この際、外部記憶装置９０に格納されたプログラム中の命令群が予めコンパイラ等により圧縮されているときは、このプログラムはそのままメモリ１０に読み込まれる。しかし、このプログラム中の命令群がまだ圧縮されていないときは、このプログラムをメモリ１０に読み込む過程において、たとえばＯＳに組み込まれバックグラウンドで走っている圧縮処理プログラムあるいはコンピュータ１００自体に組み込まれた圧縮処理ロジック回路により、プログラム中の命令群を圧縮しながら　メモリ１０にロードする。（圧縮の方法は図５を参照して後述する。）メインメモリ１０は、命令格納部１２　命令記述情報部１４　およびデータ格納部１６を含んでいる。上記プログラムの圧縮された命令はメモリ１００の命令格納部１２に格納され、命令の圧縮状態に関する情報は命令記述情報部１４に格納され、このプログラムで使用されるデータはデータ格納部１６に格納される。データ格納部１６に格納されたデータＤ１０は、適宜、データキャッシュ８０を介して、あるいは直接に、命令実行ユニット６０に供給される。

【００１４】メモリ１０にロードされた命令は圧縮されているので、命令格納部１２および命令記述情報部１４で消費されるメモリの量は、圧縮しない場合よりも少なくなっている。すなわち、命令圧縮によりメモリが節約されている。（もとのプログラムがＮＯＰを多く含めば含むほど、メモリ節約量は大きくなる。）メモリ１０にロードされた圧縮されている命令は、その実行前に元の形に復元しなければならない（そうしないと正常な並列処理が行なわれず、エラーがでる）。この命令復元を行なうために、コンピュータ１００は、命令復元部２０および命令記述復元割込発生部２２を備えている。

【００１５】すなわち、命令記述情報部１４の内容に対応して圧縮された命令の実行時に、割込発生部２２が、割り込みＤ２２を発生させる。すると命令復元部２０は、命令記述情報部１４からの情報Ｄ１４に基づき命令格納部１２からの圧縮された命令Ｄ１２を元の形に復元し、復元された命令Ｄ２０を命令キャッシュ３０（あるいは命令レジスタ群４０）に格納する。

【００１６】命令キャッシュ３０に格納された復元命令Ｄ２０は、命令レジスタ群（命令バッファ）４０の命令パイプに投入され、ここから順に命令Ｄ４０が命令デコーダ５０に送り出される。

【００１７】命令Ｄ４０はデコーダ３０によりデコードされる。デコードされた命令（適宜ＮＯＰを含む４つの並列処理命令）Ｄ５０は実行ユニット６０のフィールド＃１～＃４に同時に投入され、並列処理される。

【００１８】実行ユニット６０での処理結果Ｄ６０はデータキャッシュ８０あるいはデータ格納部１６に書き込まれる。こうして、命令キャッシュ３０あるいは命令レジスタ群４０に読み込まれた（圧縮復元後の）命令群が、実行ユニット６０のフィールド＃１～＃４において次々と並列実行される。

【００１９】図５は、図１の並列処理コンピュータが扱う命令の圧縮処理を説明するフローチャートである。こ

5　　6

の命令圧縮は、実行プログラムを生成する際のコンパイラによる処理（リンクを行なう前後の処理）でもよいし、コンピュータ100がプログラムをメモリ10にロードする時に行なう処理でもよい。また図6は、4つの命令フィールド＃1～＃4からなるVLIW命令1～5がどのように圧縮されるかの一例を示している。

【0020】ここでは、デバッグを行なうため高級言語のソースプログラム（C言語で記載されたソースコード等）をシーケンシャルな命令コードになるようにコンパイルしリンクした場合を想定している。

【0021】まず、並列実行型VLIWマシンのシーケンシャル実行用命令コード（ロードモジュール）が入力され（ステップST40）、入力された命令コード中に同時実行する命令がいくつあるかがチェックされる（ステップST42）。

【0022】例えば図6の命令1で示すように同時実行する命令が1つ（ADD）だけの場合（ステップST44、イエス）、命令記述情報部14には「1命令のみ実行」を示すとともにこの1命令の位置をビット「1」で示す情報「1000」が書き込まれる（ステップST46）。

【0023】同時実行する命令が1つの場合、命令記述情報部14の4ビット情報の各ビットの和（Σ＃1～＃4）は1であり、これが「1命令のみ実行」を示す。また命令記述情報部14の4ビット情報「1000」中の「1」の位置が同時実行する命令（ADD）のフィールド＃1を示し、「0」の位置が命令復元時に無実行命令NOPが挿入されるフィールド＃2～＃4を示す。

【0024】命令記述情報部14に「1命令のみ実行」を示す情報「1000」が書き込まれると、入力された命令コードからNOPを取り除いた命令（ADD）だけが命令格納部12に格納される（ステップST48）。こうして命令1の圧縮（NOPの除去）が終了する。

【0025】命令1は入力された命令コードの最終命令ではないので（ステップST52、ノー）次の命令2が取り込まれ（ステップST54）、その命令の同時実行命令数がチェックされる（ステップST42）。この場合も同時実行命令は1つだけ（SUB）である（ステップST44、イエス）。すると図6に示すように、命令記述情報部14には「1命令のみ実行」を示すとともにこの1命令の位置をビット「1」で示す情報「0010」が書き込まれる（ステップST46）。

【0026】命令記述情報部14に「1命令のみ実行」を示す情報「0010」が書き込まれると、入力された命令コードからNOPを取り除いた命令（SUB）だけが命令格納部12に格納される（ステップST48）。こうして命令2の圧縮（NOPの除去）が終了する。

【0027】同様に、図6の命令3に対する命令圧縮処理が行なわれる（情報部14には「1000」が書き込まれ、格納部12にはSUBが格納される）。一方、図6の命令4で示すように同時実行する命令が複数（ADDとMUL）の場合（ステップST44、ノー）、命令記述情報部14には、複数の「並列実行命令あり（つまり同時実行命令が2以上）」を示すとともにこれら複数命令の位置をビット「1」で示す情報「0110」が書き込まれる（ステップST56）。

【0028】命令記述情報部14に「複数命令同時実行」を示す情報「0110」が書き込まれると（和Σ＃1～＃4は2以上）、入力された命令4そのもの（NOP ADD、MUL、NOP）がメモリ10（命令格納部12）の所定アドレス（0X1000～0X1003）に格納される（ステップST58）。この場合は命令4の圧縮（NOPの除去）は行なわれず、命令記述情報部14には命令4を格納した場所の先頭アドレス（0X1000）が書き込まれる（ステップST60）。

【0029】あるいは、命令記述情報部14に「複数命令同時実行」を示す情報「0110」を書き込んだあと（和Σ＃1～＃4は2以上）、入力された命令4からNOPを取り除いたもの（ADD、MUL）をメモリ10の所定アドレス（0X1000～0X1001）に格納するようにしてもよい（ステップST58）。この場合は命令4の圧縮（NOPの除去）が行なわれており、命令記述情報部14には命令4の実行命令本体（ADD、MUL）を格納した場所の先頭アドレス（0X1000）が書き込まれる（ステップST60）。

【0030】なお、ステップST44でノーとなる場合（同時実行命令が1命令のみでない場合）としては、同時実行命令が複数ある場合に限定はされない。たとえばOSがスーパバイザモードで使用する特殊な長い命令（通常の命令が32ビット固定長命令である場合に特権命令が32～128ビットの可変長命令であるときなど）の場合、「1命令のみの実行」として扱かわれず（ステップST44、ノー）、この特殊命令の格納先が命令記述情報部14に格納され（ステップST56）、そのあとこの特殊命令がメモリ10の所定箇所に格納される（ステップST60）。

【0031】命令4は入力された命令コードの最終命令ではないので（ステップST52、ノー）次の命令5が取り込まれ（ステップST54）、その命令の同時実行命令数がチェックされる（ステップST42）。この場合は同時実行命令は1つだけ（ADD）であり（ステップST44、イエス）、命令記述情報部14には「1命令のみ実行」を示すとともにこの1命令の位置をビット「1」で示す情報「1000」が書き込まれる（ステップST46）。命令記述情報部14に情報「1000」が書き込まれると、入力された命令コードからNOPを取り除いた命令（ADD）だけが命令格納部12に格納される（ステップST48）。こうして命令5の圧縮（NOPの除去）が終了する。

【0032】命令5が最終命令であれば（ステップST

52．イエス）命令の圧縮処理は終了する。図5のステップST40～ST60の処理が済むと、図6の左側に示した非圧縮並列処理命令1～5（32ビット命令を4つ並列に処理する128ビットVLIW命令）は図6の右側に示すように圧縮され、圧縮された命令とその命令記述情報がメモリ10に格納される。

【0033】図5の処理の結果として命令格納部12に格納される命令コードの合計サイズは、削除したNOPの分、元の並列処理命令コードの合計サイズよりも小さくなる。命令記述情報部14のビットサイズは命令格納部12に格納されなかったNOPよりも小さいので、命令記述情報部14の存在によるメモリ10の消費量よりも、NOPを格納しなかったことによるメモリ10の節約量の方が大きい。

【0034】また、命令記述情報部14の4ビットを命令格納部12に格納される32ビット命令の一部として取り込めば、命令記述情報部14によるメモリの消費はなくなり（この場合、各命令は28ビットで記述されることになる）、NOP削除によるメモリ節約分がそのままフリーエリアとしてメインメモリ10に残る。

【0035】図2は、図1の並列処理コンピュータ100において実行される処理を説明するフローチャートである。また図3は、図2におけるNOP復元処理の一例を示すフローチャートである。この処理は、命令復元部20内のハードウエアロジックまたはコンピュータ100のマイクロコードで実行できる。

【0036】まず、圧縮された命令1（図6参照）が命令復元部20に取り込まれ（ステップST10）、引き続き取り込んだ命令1の命令記述情報部14の内容がチェックされる（ステップST12）。

【0037】4ビット情報部14の各ビット中の「1」の和Σ＃1～＃4が1ならば、圧縮された命令1は同時実行命令数が1つだけであると判定され（ステップST14．イエス）、命令1に対応する命令格納部12の内容（命令ADD）が読み出される（ステップST16）。

【0038】すると、NOP復元処理に入る（ステップST18）。すなわち図3に示すように、まず命令記述情報部14のビット1の位置に該当するフィールド＃1が抽出される（ステップST181）。続いて　図7に示すように、抽出されたビット1の位置に該当するフィールド＃1に読み出された命令ADDが配置され、残りのフィールド＃2～＃4にNOPが挿入される（ステップST182）。

【0039】こうして正規の並列処理命令（128ビットVLIW命令）に復元された命令1は、命令キャッシュ30（または命令レジスタ群40）にロードされる（ステップST20）。

【0040】命令1が最終命令でないときは（ステップST22、ノー）、次の命令2が命令復元部20に取り込まれ（ステップST24）、取り込んだ命令2の命令記述情報部14の内容がチェックされる（ステップST12）。

【0041】4ビット情報部14の各ビット中の「1」の和Σ＃1～＃4が1なので、圧縮された命令2は同時実行命令数が1つだけであると判定され（ステップST14．イエス）、命令2に対応する命令格納部12の内容（命令SUB）が読み出される（ステップST16）。

【0042】すると、命令2のNOPが復元され（ステップST18）、復元された並列処理命令2は　命令キャッシュ30（または命令レジスタ群40）にロードされる（ステップST20）。

【0043】同様にして、命令3のNOPが復元され（ステップST18）、復元された並列処理命令3は、命令キャッシュ30（または命令レジスタ群40）にロードされる（ステップST20）。

【0044】次に、非圧縮命令4（図8参照）が命令復元部20に取り込まれ（ステップST10）、取り込んだ命令4の命令記述情報部14の内容がチェックされる（ステップST12）。

【0045】ここでは4ビット情報部14の各ビット中の「1」の和Σ＃1～＃4が2なので　命令4は同時実行命令数が2つあると判定される（ステップST14．ノー）。すると図1の命令記述復元割込発生部22により割り込みが発生する（ステップST26）。

【0046】この割り込みが生じると、命令復元部20は、命令格納部12の内容（0x1000）から、命令4の格納先アドレス（0x1000から連続する4アドレス分）を算出する（ステップST28）。すると図7に示す命令4の格納先アドレス（0x1000～0x1003）から命令4の内容（NOP、ADD、MUL、NOP）が読み出される（ステップST30）。

【0047】こうして読み出された並列処理命令4は、命令キャッシュ30（または命令レジスタ群40）にロードされる（ステップST32）。なお、命令4もNOP除去の圧縮を受けている場合は（つまり4ビット情報部14の各ビット中の「1」の和Σ＃1～＃4が3以下の場合は）、命令4中の実行命令（ADD、MUL）をその格納先アドレス（図6下中央の0x1000～0x1001）から読み出してから（ステップST30）、読み出された実行命令（ADD、MUL）にNOPを付加する復元処理（図3のステップST181～ST182）を行なってもよい。

【0048】この場合、情報部14中で最初にビット「1」が立っているフィールド（＃2）に最初の格納先アドレス（0x1000）の命令（ADD）が配置され、情報部14中で2番目にビット「1」が立っているフィールド（＃3）に次の格納先アドレス（0x1001）の命令（MUL）が配置され、情報部14中ビット

9　　10

「0」のフィールド（＃1．＃4）にNOPが配置される。

【0049】命令4が最終命令でないときは（ステップST22、ノー）、次の命令5が命令復元部20に取り込まれ（ステップST24）、取り込んだ命令5の命令記述情報部14の内容がチェックされる（ステップST12）。4ビット情報部14の含むビット中の「1」の和Σ＃1～＃4は1なので、圧縮された命令5は同時実行命令数が1つだけであると判定され（ステップST14、イエス）、命令5に対応する命令語納部12の内容（命令ADD）が読み出される（ステップST16）。すると、命令5のNOPが復元され（ステップST18）、復元された並列処理命令5が、命令キャッシュ30（または命令レジスタ群40）にロードされる（ステップST20）。

【0050】命令5が最終命令であるときは（ステップST22、イエス）、命令キャッシュ30（または命令レジスタ群40）にロードされた命令1～5が順次図1の命令デコーダ50でデコードされ、デコードされた命令1～5が実行ユニット60のフィールド＃1～＃4で並列に同時処理される（ステップST34）。

【0051】図2の処理の結果、メインメモリ10に読み込まれた命令1～5が圧縮された命令（サイズ小）であっても、命令キャッシュ30あるいは命令レジスタ群40に書き込まれた実行直前の命令1～5は、図10に示すような非圧縮の復元命令となっている。命令実行ユニット60は、この復元命令1～5をフィールド＃1～＃4で並列実行する。

【0052】なお、復元された命令1～5が全て命令キャッシュ30に格納され終わるまで待つのではなく、適当な数の復元命令がキャッシュ30（又はレジスタ40）に溜まったら実行ユニット60で命令実行（ステップST34）を開始するようにしてもよい。この場合はステップST22の前に命令実行ステップが挿入される。

【0053】また、命令キャッシュ30あるいは命令レジスタ群40に書き込まれた復元命令1～5はその実行後は消滅してもエラーは生じない。したがって、復元された該読命令（命令6以降）を次々と命令キャッシュ30あるいは命令レジスタ群40に書き込んで、命令キャッシュ30あるいは命令レジスタ群40内の実行済み旧命令（命令1～5）をどんどん消去することができる。このため命令キャッシュ30あるいは命令レジスタ群40が多数の復元命令でオーバーフローしてエラーを出すことはない。

【0054】もし、コンピュータ100が4キロバイトの命令キャッシュ30を備えており、実行ユニット60のフィールド＃1～＃4で実行される各命令が32ビット（4バイト）固定長であるとすれば、キャッシュ30は復元後の32ビット命令を最大1000個待つことができる。コンピュータ100で処理しようとするプログラムモジュール中の命令数が1000個以内（4ワード構成のVLIW命令で数えれば250個以内）ならば、復元後の非圧縮命令は全て命令キャッシュ30に収まってしまう。この場合は上記復元命令のオーバーフローは生じない。

【0055】必要な復元命令が全て命令キャッシュ30に収まったあとは、もはや命令復元処理は不要となるから、命令復元のためにコンピュータ100全体の処理速度が落ちることはない。このことから、コンピュータ100で実行されるプログラム中の命令数に対して、命令キャッシュ30は十分な記憶容量を持っていることが望ましい。

【0056】なお、コンピュータ100が汎用レジスタ（または命令レジスタ）を豊富に持っており、これらのレジスタ中に必要な復元命令の大部分を保持できるなら、大容量の命令キャッシュ30は必ずしも必要ではない。

【0057】図6の例では命令記述情報部14を命令実行ユニット60のフィールド数と同数のビット情成とし、圧縮命令のフィールド位置情報も情報部14に含ませている。このため図8に示すような圧縮前の命令1～5が圧縮された後これを復元するど、図10に示すように元通りの命令1～5が得られる。

【0058】ここで、命令1のようにVLIW命令が実行命令を1つしか含まないときは、この実行命令（ADD）がフィールド＃1～＃4のどこで実行されても、ソフトウエア上は、その処理結果は同じになる（ハードウエア上では、同一フィールドでの連続命令実行に伴いハザードの問題が起きる可能性がある。ここではハザードは起きないと仮定する）。この場合、命令記述情報部14は必ずしも実行命令（ADD）のフィールド位置情報を含んでいる必要はない。

【0059】そのような場合では、命令記述情報部14は「VLIW命令が実行命令を1つしか含まない」かどうかを区別する情報だけを持てばよく、情報部14を1ビットフラグで構成することができる。

【0060】図8は、命令記述情報部14を1ビットフラグで構成した場合に、VLIW命令がどのように圧縮されるかを説明するための図である。すなわち、命令1では実行命令が1つだけであるから（図5のステップST44、イエス）、情報部14のフラグが「0」とされ（ステップST46）、命令格納部12に実行命令（ADD）が格納される（ステップST48）。

【0061】一方、命令4では実行命令が1つだけではないから（ステップST44、ノー）、情報部14のフラグが「1」とされ（ステップST56）、命令格納部12に命令4の格納先先頭アドレス（0x1000）が格納される（ステップST58）。このあとこの先頭アドレスから連続する4アドレス（0x1000～0x1

11　　12

０９３）に命令４がそのまま格納される（ステップＳＴ６９）。

【００６２】図４は、図８の例におけるＮＯＰ復元処理の例を示している。すなわち、命令１の情報部１４のフラグが「０」である場合（図２のステップＳＴ１６）、格納部１２から実行命令（ＡＤＤ）を取り出してこれを所定フィールド．たとえばフィールド＃１に置く。しかる後に残りフィールド＃２～＃４をＮＯＰで埋めて（ステップＳＴ１８）、図９に示すように命令１を復元する。復元された命令１は、命令キャッシュ３０または命令レジスタ４０に転送される（ステップＳＴ２０）。

【００６３】一方、命令４の情報部１４のフラグが「１」である場合、命令復元割込を発生させる（図２のステップＳＴ２６）。続いてメモリ１０のアドレス０ｘ１０００～０ｘ１００３を読出し（ステップＳＴ２８）、そのアドレスから実行命令（ＮＯＰ、ＡＤＤ、ＭＵＬ、ＮＯＰ）を取り出して（ステップＳＴ３０）、これをフィールド＃１～＃４に配置する。こうして得られた図９に示すような命令４は、命令キャッシュ３０または命令レジスタ４０に転送される（ステップＳＴ３２）。

【００６４】図８のように命令記述情報部１４が１ビット構成の例では、ＶＬＩＷ命令中の実行命令が１つだけの場合、実行命令（命令１のＡＤＤ．命令２のＳＵＢなど）のフィールド位置を特定できない。したがって、この場合の命令復元処理においては、命令復元に一定の規則を設けておく必要が生じる。

【００６５】図１１は、第１の命令復元規則にしたがって復元されたＶＬＩＷ命令群の例である。ここでは、フィールド＃１に１クロック処理命令（ＡＤＤ、ＳＵＢ）を集め、フィールド＃２にＮＯＰまたは２クロック処理命令（ＭＵＬ）を集め、フィールド＃３～＃４にＮＯＰまたは図示しない３クロック以上の処理命令を集めている。この例ではフィールド＃１での逐次命令処理においてハザードが生じないことを約束している。

【００６６】図１２は、第２の命令復元規則にしたがって復元されたＶＬＩＷ命令群の例である。ここでは、フィールド＃１にＮＯＰまたは第１の命令（ＡＤＤ）を集め、フィールド＃２にＮＯＰまたは第２の命令（ＳＵＢ）を集め、フィールド＃３にＮＯＰまたは第３の命令（ＭＵＬ）を集め、フィールド＃４にＮＯＰまたは図示しない第４の命令（割り算命令ＤｉＶなど）を集めている。

【００６７】図１３は、第３の命令復元規則にしたがって復元されたＶＬＩＷ命令群の例である。ここでは、４命令（命令１～４）処理を１サイクルとし、各フィールドは１サイクル中で１回はＮＯＰ以外の命令を実行するようにしている。

【００６８】なお前述した実施例においては、４ビット（または１ビット）命令記述部１４は３２ビット命令格納部１２の他に用意されているが、この命令記述部１４は３２ビット命令格納部１２の一部に組み込んでもよい。たとえば３２ビット中４ビット（または１ビット）を命令記述部１４のために用い、残り（２８ビット～３１ビット）をＡＤＤ、ＳＵＢなどの命令記述のために用いるようにしてもよい。

【００６９】上述した実施例によれば、ＶＬＩＷ型コンピュータでシーケンシャルなコードを実行する場合に命令記述部分を圧縮できるので、大規模なプログラムのシーケンシャルな実行を少ない実メモリで実行できる。

【００７０】この際、並列度の高い命令（デバック時のシーケンシャルコードではＮＯＰが多くなる）ほどサイズ圧縮効果は高い。なお、この発明は、ＶＬＩＷ型に限らず、命令コード中に適宜ＮＯＰが挿入されたプログラムを実行時にメモリに読み込む他方式の並列処理型コンピュータ（スーパースカラ型コンピュータ等）にも応用可能である。

【００７１】

【発明の効果】この発明によれば、メモリにロードされる並列処理プログラムの命令コードは圧縮されているから、圧縮された分だけメモリのフリーエリアは広くなっている。圧縮された命令は、その実行の度に本来の命令に逐次復元される。このため、並列処理命令の実行に差し障りはない一方で、広いフリーエリアがメモリに確保できる。換言すれば、並列処理コンピュータおける大規模プログラムのシーケンシャルな実行を、より少ないメモリで行なうことができる。

【図面の簡単な説明】

【図１】図１は、この発明の一実施例に係るＶＬＩＷ型並列処理コンピュータの構成を示すブロック図。

【図２】図２は、図１の並列処理コンピュータにおいて実行される処理を説明するフローチャート。

【図３】図３は、図２におけるＮＯＰ復元処理の一例を説明するフローチャート。

【図４】図４は、図２におけるＮＯＰ復元処理の他例を説明するフローチャート。

【図５】図５は、図１の並列処理コンピュータが扱う命令の圧縮処理を説明するフローチャート。

【図６】図６は、４つの命令フィールドからなるＶＬＩＷ命令がどのように圧縮されるかの一例を説明する図。

【図７】図７は、図６の例において圧縮されたＶＬＩＷ命令がどのように復元されるかを説明する図。

【図８】図８は、４つの命令フィールドからなるＶＬＩＷ命令がどのように圧縮されるかの他例を説明する図。

【図９】図９は、図８の例において圧縮されたＶＬＩＷ命令がどのように復元されるかを説明する図。

【図１０】図１０は、４つの命令フィールドからなるＶＬＩＷ命令がオリジナル通りに復元された場合を例示する図。

【図１１】図１１は、４つの命令フィールドからなるＶ

13

LIW命令が第1の所定規則（フィールド1に1クロック処理命令を集める）に従い復元された場合を例示する図。

【図12】図12は、4つの命令フィールドからなるVLIW命令が第2の所定規則（同じ演算命令は同じフィールドに集める）に従い復元された場合を例示する図。

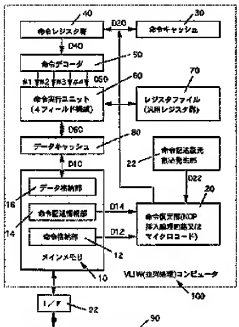
【図13】図13は、4つの命令フィールドからなるVLIW命令がモディファイされて復元された場合を例示する図。

14

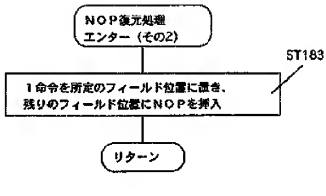
＊【符号の説明】

10…メインメモリ、12…命令格納部、14…命令記述情報部、16…データ格納部、20…命令復元部、22…命令記述復元完了発生部、30…命令キャッシュ、40…命令レジスタ群（命令バッファ）、50…命令デコーダ、60…命令実行ユニット（4並列処理）、70…レジスタファイル（汎用レジスタ群）、80…データキャッシュ、90…外部記憶装置、92…インターフェイス、100…VLIW型並列処理コンピュータ。

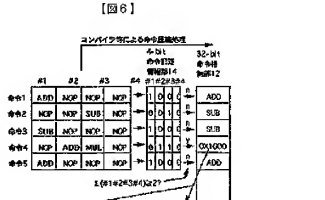
【図1】



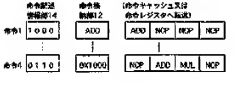
【図4】



【図6】

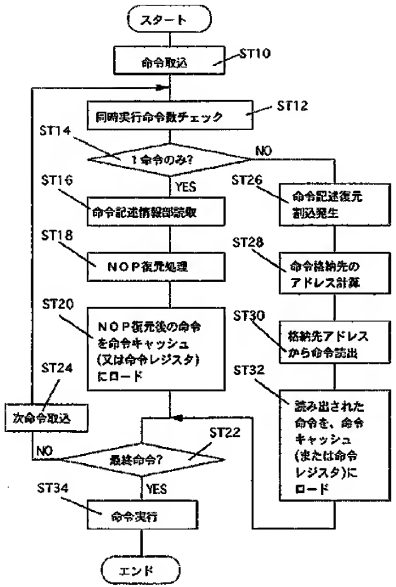


【図7】



【図9】

【図2】



【図10】

	#1	#2	#3	#4
命令1	ADD	NOP	NOP	NOP
命令2	NOP	NOP	SUB	NOP
命令3	SUB	NOP	NOP	NOP
命令4	NOP	ADD	MUL	NOP
命令5	ADD	NOP	NOP	NOP

オリジナル通りに復元した場合

【図11】

	#1	#2	#3	#4
命令1	ADD	NOP	NOP	NOP
命令2	SUB	NOP	NOP	NOP
命令3	SUB	NOP	NOP	NOP
命令4	ADD	MUL	NOP	NOP
命令5	ADD	NOP	NOP	NOP

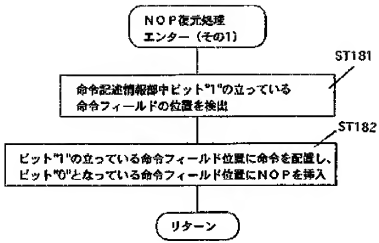
1クロック命令はフィールド1に集中

【図12】

	#1	#2	#3	#4
命令1	ADD	NOP	NOP	NOP
命令2	NOP	SUB	NOP	NOP
命令3	NOP	SUB	NOP	NOP
命令4	ADD	NOP	MUL	NOP
命令5	ADD	NOP	NOP	NOP

同じ演算は同じフィールドに集める

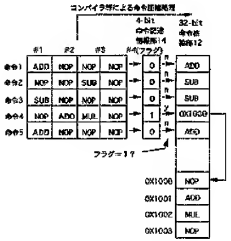
【図3】


NOP復元処理
エンター（その1）
ST181
命令記述情報部中ビット"1"の立っている
命令フィールドの位置を検出
ST182
ビット"1"の立っている命令フィールド位置に命令を配置し、
ビット"0"となっている命令フィールド位置にNOPを挿入
リターン

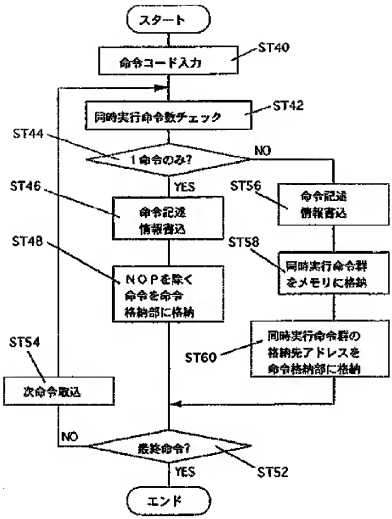
【図13】


#1 #2 #3 #4
命令1 ADD NOP NOP NOP
命令2 NOP NOP SUB NOP
命令3 NOP NOP NOP SUB
命令4 NOP ADD MUL NOP
命令5 ADD NOP NOP NOP
オリジナルモヂィファイルで復元

【図8】


コンパイラ等によるNOP圧縮処理
4-bit 命令記述情報部11
32-bit 命令語 語長12
#1 #2 #3 #4フラグ
命令1 ADD NOP NOP NOP 0
命令2 NOP NOP SUB NOP 0
命令3 SUB NOP NOP NOP 0
命令4 NOP ADD MUL NOP 1
命令5 ADD NOP NOP NOP 0
フラグ＝1？
ADD
SUB
SUB
0X1000
ADD
0X1000 NOP
0X1001 ADD
0X1002 MUL
0X1003 NOP

【図5】


スタート
命令コード入力
ST40
同時実行命令数チェック
ST42
ST44
1命令のみ?
NO
YES
ST46
命令記述
情報書込
ST56
命令記述
情報書込
ST48
NOPを除く
命令を命令
格納部に格納
ST58
同時実行命令群
をメモリに格納
ST54
次命令取込
ST60
同時実行命令群の
格納先アドレスを
命令格納部に格納
NO
最終命令?
YES
ST52
エンド